Python Bootcamp & Masterclass

lambda functions

gknxt

I have never considered Python to be heavily influenced by functional languages, no matter what people say or think. I was much more familiar with imperative languages such as C and Algol 68 and although I had made functions first-class objects, I didn't view Python as a functional programming language.

- Guido van Rossum

Models of Computation

1

Functional languages get their origin in mathematical logic and lambda calculus. They adopt a state-less, declarative approach of programming that emphasizes abstraction, data transformation, composition, and purity.
e.g: Lisp

2

Imperative programming languages embrace Alan Turing's state-based model of computation This style consists of programming with statements, driving the flow of the program step by step with detailed instructions.
e.g: Java, C, Python

Python is not a functional language, but it adopted some functional concepts - map(), reduce(), filter() and the lambda expression - were taken from the functional paradigm and were added to the language.

Lambda expressions in Python have their roots in lambda calculus, a model of computation invented by Alonzo Church in 1930

Why to create a function if we want use it just once? You probably don't want to buy metal forks, knives, and spoons for a casual picnic; rather, you can just buy plasticware. An anonymous function can be used for passing to other functions. It goes away, removed from memory as soon as it's no longer needed.

Often, a small function needs to be passed to another function, like the key function used by a list's sort method. In such cases, a large function is usually unnecessary, and it would be awkward to have to define the function in a separate place from where it's used.

A **lambda function or lambda expression**

is an anonymous (unnamed) function with no return statement as the value of the expression is automatically returned

```
lambda <parameter(s)>:<return_exp>
```

`lambda`	The keyword that informs Python that what follows is a lambda function
`parameter(s)`	(optional) A comma-separated list of parameters as input to lambda
return_exp	return expression
	(A lambda function can’t contain any statements)

```
(lambda x, y: x + y)(4,6)
```

```
10
```

gk nxt

Online Resources

For best python resources, please visit:

gknxt.com/python/

gknxt

Python Bootcamp & Masterclass

Thank You

for your Rating & Review

gk nxt